1968

An application of graph coloring to a scheduling problem

Mack, Jacob A.

Monterey, California. Naval Postgraduate School

AN APPLICATION OF GRAPH COLORING TO A SCHEDULING PROBLEM

JACOB A. MACK, III

AN APPLICATION OF GRAPH COLORING TO A SCHEDULING PROBLEM

by

Jacob A. MACK, III
Lieutenant, United States Navy
B.S., United States Naval Academy, 1961

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
WITH MAJOR IN MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
June 1968

ABSTRACT

A class scheduling problem is formulated as a graph coloring problem. A computer program based on an algorithm recently developed by Welsh and Powell, Computer Journal, Vol. 10, May 1967, pp. 85-86, is used to obtain a solution to the coloring problem. While the program fails to provide an acceptable schedule in this application, the results indicate that improvements in the coloring algorithm may yield acceptable schedules.

TABLE OF CONTENTS

SECTION	TITLE	PAGE
	Abstract	2
	List of Illustrations	5
1.	Introduction	7
2.	Definition of the Problem	11
	A. Preliminary Definitions	11
	B. Definition of the Scheduling Problem	13
3.	The Coloring Process	26
	A. The Algorithm	26
	B. Coloring the Graph by Vector Colors	31
4.	Conclusions	37
	Bibliography	39
	Table 1	41
	Appendix	42

LIST OF ILLUSTRATIONS

FIGURE	TITLE	PAGE
1	Example Data Graph	14
2	Example Conflict Graph	15
3	Data Graph	19
4	Hourly Conflict Graph	21
5	Daily Conflict Graph	22
6	Sequence Graph	23
7	An Arbitrary Graph	28
8	Relabeled Arbitrary Graph	29
	Flowchart A	42
	Flowchart B	43
	Flowchart C	44
	Flowchart D	46
	Flowchart E	47
	Flowchart F	48
	Flowchart G	49
	Flowchart H	50
	Flowchart I	51
	Flowchart J	52
	Flowchart K	53
	Flowchart L	54

1. Introduction.

The procedure of scheduling classes varies with each academic institution. In most instances schools announce beforehand a list of courses and times they are to be offered. In this case students schedule themselves to courses which do not conflict. A different scheduling problem occurs in the case where students are required to take predetermined courses. Here the problem is to schedule the courses so that students have no conflicts. The problem of scheduling classes at the Naval Postgraduate School, Monterey, California, falls in this category. It is this problem which is investigated here.

The digital computer has been used as an aid in scheduling before. Appleby, Blake, and Newman [1] attempt to produce school time-tables using a general-purpose digital computer. Their suggested methods of approach include

(1) a random trial solution updated to eliminate conflicts,

(2) a trial of all combinations possible until one with no conflicts is found,

(3) a random buildup of entries into a schedule, adding entries only when conflicts do not occur, modifying previous entries to make new entries possible, and

(4) a heuristic approach.

The importance of knowing whether or not a solution does exist is stressed. The last of three successively more

complex programs is estimated to speed up scheduling by a factor of 200 to 400 over the hand operation.

Csima and Gottlieb [6] describe a time-table construction technique which uses matrices of zeroes and ones to produce results in the limited situations attempted. In real situations where preassignments or special constraints appear, further progress must be made. It is also hoped that with refinements in the program the computer execution time can be reduced.

A method of determining examination time-tables has been described by Cole [5]. Citing Sherman's [9] pessimism in finding a schedule for a large school, he emphasizes producing an acceptable solution for as many courses as possible with reasonable computer time. Dependent on the imposed constraints, the execution time on the small-store computer was estimated to be several hours if all of 340 courses were considered.

Another method using GASP, Generalized Academic Simulation Program, a program written at M.I.T. [7], approaches the problem by assigning students to a previously determined course schedule.

This paper will treat the scheduling problem by assigning days and hours to courses, so that the constraints imposed by the students as well as instructors are satisfied.

At the Naval Postgraduate School there are aspects to the problem which make it different from most institutions. These aspects are treated as restrictions to the basic scheduling problem and will have an effect on its solution. For instance, the government allots a given length of time to students to follow a prescribed curriculum of study. Since certain courses are taught in sequence, students are required to take certain courses at certain times. The students' schedule thus loses some flexibility. Another aspect to be considered is the fact that all naval aviators must be scheduled either a complete morning or afternoon for weekly aircraft flights. Other restrictions include days off for instructor research and allowances made for faculty and staff education at the school itself.

A graph theoretic approach, which makes use of the computer for the necessary calculations, is used to attempt the solution to the scheduling problem. A graph theoretic model of the scheduling problem is constructed. The problem of scheduling courses without conflicts is shown to be equivalent to a graph coloring problem. A digital computer program is produced to solve the coloring problem.

Section 2 of this paper defines the problem investigated. Preliminary definitions and terminology are first presented in graph theoretic terms. The graph theoretic model is then constructed with these tools. In section 3 an algorithm for graph coloring is analyzed. This algorithm then is used to construct a coloring program.

The results of this program are displayed in Table 1. Observations and conclusions based on these results are found in Section 4. The Appendix contains both general and detailed flowcharts, which explain the coloring program used.

2. Definition of the Problem.

A. Preliminary Definitions.

We wish to consider the basic problem of scheduling from a graph theoretic standpoint. But first we need the following basic definitions taken from Berge [3], Chapters 1 and 2.

Definition 1. Given the sets X and Y. A function Γ, mapping X into Y, is multi-valued, if to each $x \in X$ there is a well defined subset $\Gamma x \subseteq Y$. Note: Γx may be the null set.

Definition 2. Γx is said to be the set of followers of x.

Definition 3. A pair denoted by $G=(X,\Gamma)$ is said to be a graph whenever

(1) X is a set,

and (2) Γ is a multi-valued function mapping X into X. It is customary to represent the elements of the set X as points in a plane.

Definition 4. If $G=(X,\Gamma)$ is a graph and $x \in X$, then x is said to be a point, vertex, or node of the graph G.

Definition 5. If $x \in X$ and $y \in X$ such that $y \in \Gamma x$, the ordered pair (x,y) is said to be an arc of the graph G. Arcs of a graph are usually represented by an arrow from the first to the second element of the ordered pair.

In case the order in which the vertices appear in the ordered pair (x,y) has no significance, we denote the pair by $[x,y]=[y,x]$. Such an unordered pair is called an edge of the graph G.

An edge of the graph is usually represented by a curve joining two vertices.

Definition 6. If $G=(X,\Gamma)$ is a graph and $U=\{[x,y] : y\in \Gamma x \text{ for all } x\}$ the set of edges of G, then an alternate notation for the graph is the pair (X,U). Since this graph contains only undirected edges, it is sometimes referred to as an undirected graph.

Associated with every graph is a matrix. We shall define it as follows:

Definition 7. If G=(X,U) is a graph, then $A=(a_{ij})$ is called the matrix associated with G if a_{ij} equals the number of edges from node x_i to node x_j.

Note: A is a square symmetric matrix.

Definition 8. Given the graph (X,U), where $x\in X$ and $y\in X$, $x\neq y$, x and y are said to be adjacent if $[x,y]=[y,x]\in U$.

Definition 9. If G=(X,U) is a graph and if $x\in X$, the number of edges having an endpoint at x is said to be the degree of x.

B. Definition of the Scheduling Problem.

Definition 10. A section is a group of students all of whom take the same courses.

Sections are used to simplify the scheduling problem. Determining a single section schedule is easier than finding a separate schedule for each individual.

Denote sections as s_i, $i=1,\ldots,n_1$, courses as c_j, $j=1,\ldots,n_2$, and professors as p_k, $k=1,\ldots,n_3$. We shall consider the set $X=\{s_i\}\cup\{c_i\}\cup\{p_i\}$ as a set of nodes for a graph. We shall also consider the sets of all pairs $U_1=\{[s_i,c_j]$: section s_i must take course $c_j\}$ and $U_2=\{[c_j, p_k]$: instructor p_k is assigned to teach course $c_j\}$. Define $U=U_1\cup U_2$. We see that U with the set X defines the graph (X,U).

Definition 11. The graph G=(X,U) described above is called the data graph.

We shall consider an example with the following conditions:

(1) Section s_1 is required to take courses c_1,c_2, and c_3.

(2) Section s_2 is required to take courses c_3 and c_4.

(3) Instructor p_1 is assigned to teach courses c_1 and c_3.

(4) Instructor p_2 is assigned to teach courses c_2 and c_4.

Then as defined above, $X=\{s_1,s_2,c_1,c_2,c_3,c_4,p_1,p_2\}$ and $U=\{[s_1,c_1],[s_1,c_2],[s_1,c_3],[s_2,c_4],[s_2,c_3],[p_1,c_1],[p_1,c_3],[p_2,c_2],[p_2,c_4]\}$. The data graph $G=(X,U)$ is illustrated as follows in Figure 1.

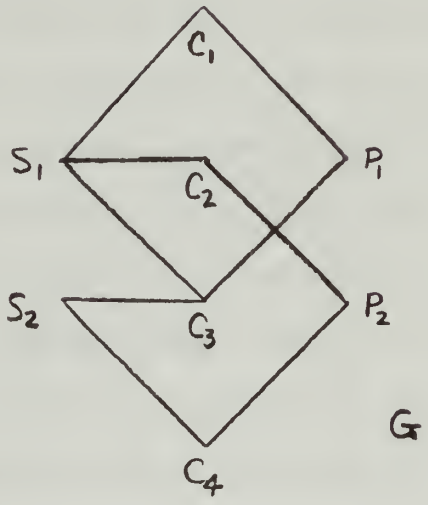


Fig. 1. Example Data Graph

Definition 12. Given the data graph $G=(X,U)$ described above, a conflict is said to occur between classes c_i and c_j, $i \neq j$,

if (1) $[s_k,c_i] \in U$ and $[s_k,c_j] \in U$ for some k,

or if (2) $[c_i,p_l] \in U$ and $[c_j,p_l] \in U$ for some l.

This is the same as saying these classes c_i and c_j cannot be scheduled simultaneously because:

(1) the same section s_k must take both classes and could not be present at two places simultaneously, or

(2) the same instructor p_l must teach both classes and could not do so simultaneously.

We consider now the concept of a conflict graph deduced from the data graph.

Definition 13. Given the data graph (X,U), a new graph (C,U') is called a conflict graph if $C=\{c_i : c_i$ is a node corresponding to class c_i for all $i\}$ and $U'=\{[c_i,c_j]:$ classes c_i and c_j have a conflict$\}$. Note: c_i cannot have a conflict with itself; therefore, for all i, $[c_i,c_i] \notin U'$.

The associated matrix with the conflict graph we shall call the conflict matrix. The conflict matrix $M=(m_{ij})$, $i,j=1,\ldots,n_2$, is such that

$$m_{ij}=\begin{cases}0, \text{ if } c_i \text{ and } c_j \text{ have no conflict.}\\ 1, \text{ if } c_i \text{ and } c_j \text{ have a conflict.}\end{cases}$$

If we consider the same example as before, we can display the conflict graph in Figure 2 below.

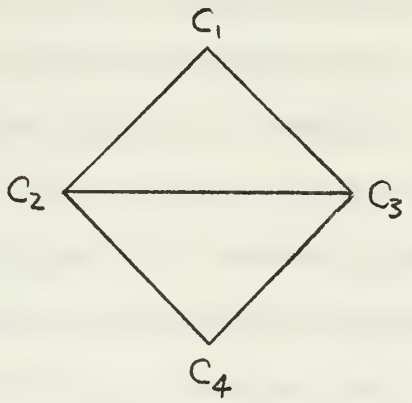


Fig. 2. Example Conflict Graph

The conflict matrix in this case is $M=\begin{pmatrix}0&1&1&0\\1&0&1&1\\1&1&0&1\\0&1&1&0\end{pmatrix}$

Coloring a graph is a process of assigning colors to each node.

Definition 14. Given an integer p, a graph G is said to be p-chromatic, if the nodes can be colored with p distinct colors so that no two adjacent nodes are assigned the same color.

Definition 15. The smallest number p for which a graph is p-chromatic is called the chromatic number of the graph G and is written k(G).

Given a group of classes with arbitrary conflicts among them, we solve the problem of scheduling by assigning time periods to these classes. Where a conflict arises between two classes we require that they be assigned different time periods. On the other hand, if we consider these classes as nodes of a graph and their conflicts as edges, the problem of coloring the nodes of this graph is seen to be equivalent to the above assignment of time periods.

The problem posed here is not necessarily one of determining the chromatic number of a conflict graph. Our problem is to find a suitable coloring of the conflict graph, which is consistent with imposed restrictions which are now discussed.

The Naval Postgraduate School uses the quarter system for teaching courses. Each course lasts twelve weeks. During the twelfth week of each quarter final exams are

held. The first eleven weeks have the same schedule. Normally each course will meet several hours each week depending upon the nature of the course.

Earlier we assigned a node c_i to each course. We shall now introduce nodes with two subscripts to differentiate between the hours of a given course. Consider a three hour course c_i. Then the hours or nodes associated with these hours are denoted $c_{i,1}$, $c_{i,2}$, $c_{i,3}$, or simply $c_{i,j}$, $j=1,2,3$. If a course is a one hour course, we shall let $j=0$.

In some cases, if there is a large number of students taking the same course in a given quarter, it may be necessary to divide this group into two or more subgroups for teaching purposes. These subgroups are assigned to take different sessions of the same course. These different sessions we shall call segments of that course. We shall introduce nodes with three subscripts to differentiate between segments of a course. In the example above, if course c_i has two segments, then the hourly nodes associated with the course are $c_{i,j,k}$, i denoting the course, j being the hour, and k identifying the segment. In this case they are $c_{i,j,k}$, $j=1,2,3$, $k=1,2$. If a course only has one segment we shall let $k=0$.

Now we let $X=\{s_i\}\cup\{c_{j,k,l}\}\cup\{p_m\}$ for all i, j, k, l, and m. We also consider $U_1=\{[s_i,c_{j,k,l}]$: section s_i must attend hour $c_{j,k,l}\}$ and $U_2=\{[c_{j,k,l},p_m]$: instructor

p_m is assigned to teach hour $c_{j,k,l}\}$. If $U=U_1 \cup U_2$, then we have $G=(X,U)$ as our data graph. We assume that as the need for segments arises, the appropriate section to segment and segment to professor assignments are made by the academic departments. We may thereafter regard these segments as different courses.

Given our new data graph, we define conflicts in exactly the same manner as in Definition 12, except that we now allow course nodes to have three subscripts.

We now consider the previous example with the modifications below.

(1) Let s_1 take courses c_1, c_2, and the first segment of course c_3.

(2) Let s_2 take course c_4 and the second segment of course c_3.

(3) Courses c_1, c_2, and c_3 will all be three hour recitation courses.

(4) Course c_4 will consist of three recitation hours and two laboratory hours to be scheduled together.

(5) Instructor p_1 will teach both segments of c_3. In this case, $X=\{c_{1,i,0}, c_{2,i,0}, c_{3,i,j}, c_{4,k,0} \cdot i=1,2,3, j=1,2, k=1,\ldots,5\}$. Figure 3 illustrates G, the data graph.

Department meetings, instructor research time, aviator flight time, and other such periods imposing constraints are displayed on the conflict graphs. In

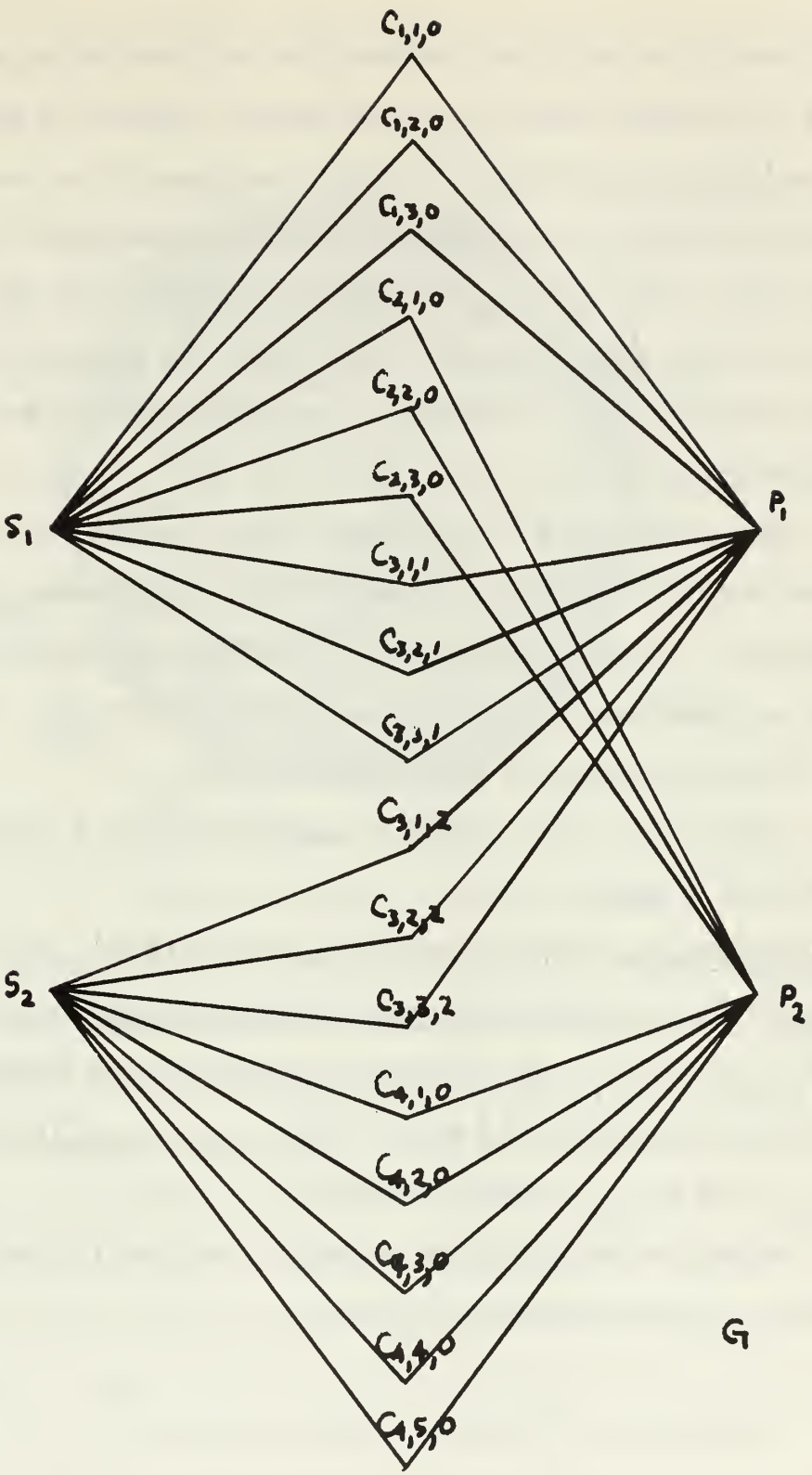


Fig. 3. Data Graph

the case of a one hour lecture for all students, we add to X a lecture node $c_{1,0,0}$ for some 1, and to U we add edges $[c_{1,0,0}, s_i]$ for all i. In the case of a one hour meeting for all professors in a given department, we add to X a node $c_{k,0,0}$ for some k, and to U we add the set of edges $\{[c_{k,0,0}, p_i]$: instructor p_i belongs to that department$\}$. Similarly other constraints are represented.

Our data graph is G=(X,U). X is the set of all nodes denoting sections, professors, courses and their segments, flights, instructor research, and special hours such as department meetings or special lectures. The set U consists of all applicable edges.

Using the above data we must introduce a slightly different concept.

Definition 16. Given the data graph G=(X,U), a new graph G'=(C,U') is called an hourly conflict graph for G, if $C=\{c_{i,j,k}$: $c_{i,j,k}$ is a node corresponding to course c_i, hour j, and segment k$\}$ and $U'=\{[c_{i,j,k}, c_{l,m,n}]$: hours $c_{i,j,k}$ and $c_{l,m,n}$ have a conflict$\}$.

Based on our previous example the hourly conflict graph is illustrated in Figure 4.

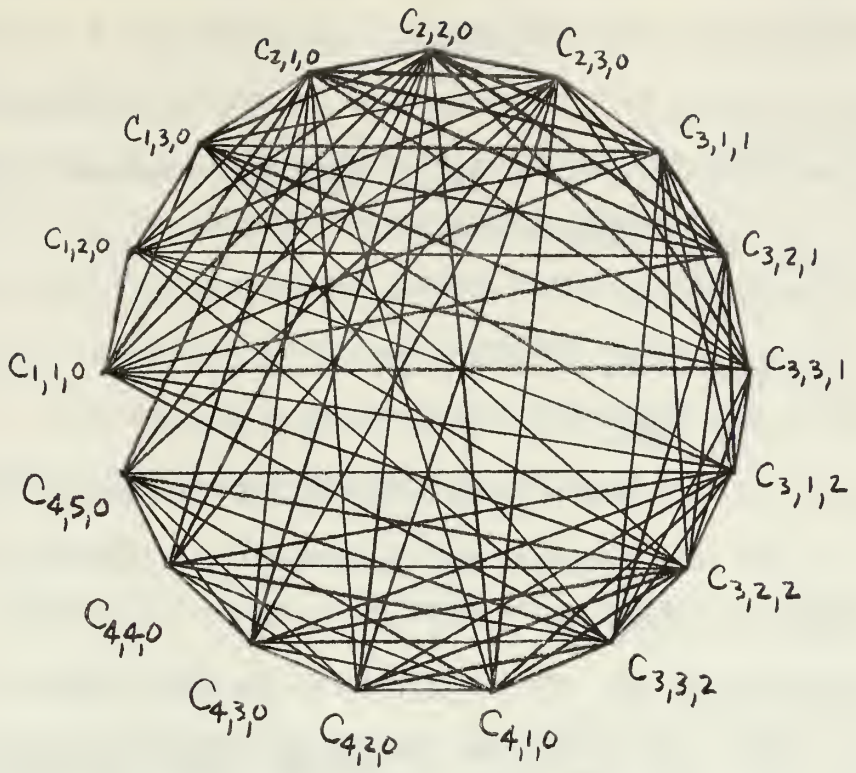


Fig. 4. Hourly Conflict Graph

Given a course that meets several times a week, we consider now conflicts between the hours of the course described above. These hours may be divided into recitations or laboratory work. For a given course, using the second subscript, we shall number the recitation hours first. For instance, a non-segmented course c_i, a five hour course consisting of three recitations, will have its hours denoted $c_{i,1,0}$, $c_{i,2,0}$, $c_{i,3,0}$, $c_{i,4,0}$, and $c_{i,5,0}$, the first three of which correspond to recitations. Note: $[c_{i,j,0}, c_{i,k,0}] \in U'$ for all $j \neq k$.

We shall restrict our problem somewhat by allowing only one recitation per course to be held daily. This brings about the following definition:

Definition 17. If C is the set of all course hour nodes, then there exists a daily conflict between $c_{i,j,k}$ and $c_{l,m,n}$ provided:

(1) i=l,

(2) k=n,

(3) $j \neq m$,

and (4) j and m both correspond to recitation hours.

We now augment our information regarding conflicts by the following:

Definition 18. If G=(X,U) is the data graph, then G''=(C,U'') is called the daily conflict graph for G, if C is the set of all course hour nodes and $U''=\{[c_{i,j,k}, c_{i,m,k}]: c_{i,j,k}$ and $c_{i,m,k}$ have a daily conflict$\}$.

If we consider our previous example again, we see that Figure 5 below illustrates G'', the daily conflict graph.

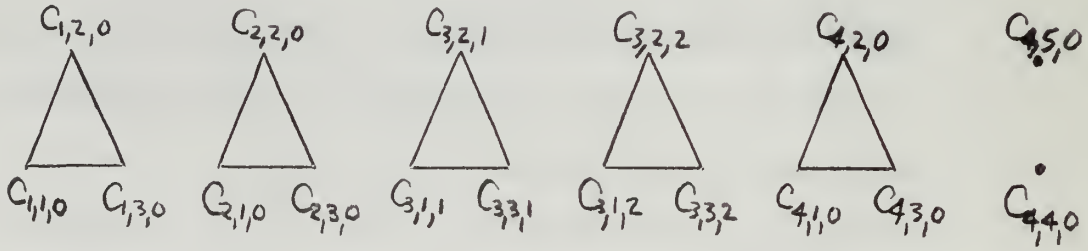


Fig. 5. Daily Conflict Graph

In the case of laboratory hours of a course or flight hours for aviators, these hours must be scheduled together during a given day. To consider this we introduce the following:

Definition 19. If G=(X,U) is the data graph, then S=(C,U''') is called the sequence graph for G, if C is the set of all course hour nodes and $U'''=\{[c_{i,j,k},c_{l,m,n}]: c_{i,j,k}$ must be scheduled together with $c_{l,m,n}\}$.

The sequence graph S for our previous example is shown in Figure 6 below.

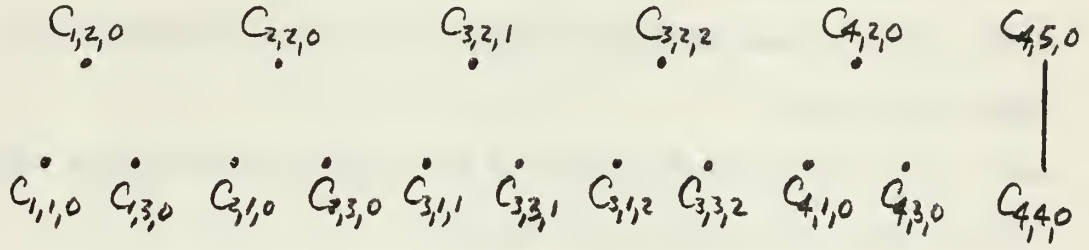


Fig. 6. Sequence Graph

We have a data graph which summarizes our input information containing all section, course, and instructor assignments. The hourly conflict graph G' displays course hours which must be scheduled on different hours. The daily conflict graph G'' displays course hours which must be scheduled on different days. Our sequence graph S indicates which course hours must be scheduled together in block form. The problem is to schedule these course hours in a five day week. The school allows nine class hours per day.

We will now consider the concept of a color vector with two components, the first of which will be a color corresponding to the day. The second will correspond to the hour within the day. Our problem is to color the course nodes of the hourly conflict graph G' so that no two adjacent course nodes have the same color vector. The coloring of the nodes of G' may be thought of as a simultaneous coloring of all three graphs G', G'', and S subject to the following restrictions:

(1) no two adjacent nodes of G' are assigned the same color vector,

and (2) no two adjacent nodes of G'' are assigned the same day color,

and (3) adjacent nodes of S are assigned the same day colors and sequential hour colors.

Note: It is sufficient for two color vectors to be different when either or both of the corresponding components are different. In view of this it is interesting to note that satisfying (2) above automatically satisfies (1) for the same two nodes.

In this coloring process we are restricted to five day colors since the schedule desired is a weekly schedule. In addition, the number of hour colors within a day must be restricted to nine. Given a satisfactory coloring of the conflict graph, we assign days to the

day colors and hours to the hour colors. The resulting schedule meets all the constraints imposed provided we have not exceeded the maximum number of colors allowable in each case.

3. The Coloring Process.

A. The Algorithm.

Although we cannot be certain of obtaining the chromatic number for a given arbitrary graph, coloration solutions may be attained by trial and error methods or by linear programming techniques [3]. An algorithm for coloring a graph, introduced by Welsh and Powell [10], has been used to attempt to solve the vector color problem of the Naval Postgraduate School course graph. This algorithm is described as follows:

Let $G=(X,U)$ be an arbitrary graph where X is the set of nodes and U the set of edges. Denote the vertices $X=\{A_i,\ i=1,\ldots,n\}$. Let the degree of the vertex A_i be denoted by d_i, where $i=1,\ldots,n$. Assume that the vertices have been numbered so that their degrees form a non-increasing sequence $d_1 \geq d_2 \geq \ldots \geq d_n$. We now define $T_i \subset X$, where $i=1,\ldots,t$. T_1 is defined as follows:

(1) $A_1 \in T_1$,

and (2) if $\{A_{i_k}\}_{k=1}^{m} \in T_1$, where $i_1=1$ and $i_1 < i_2 < \ldots < i_m$ for some m, then $A_j \in T_1$, if and only if,

(a) $j > i_m$,

and (b) $[A_j, A_{i_k}] \notin U$ for $k=1,\ldots,m$.

Note: $T_1 \neq \emptyset$.

Similarly we define T_2, where $T_2 \subset (X-T_1)$. T_2 is constructed as follows:

(1) If i is the least integer for which $A_i \notin T_1$, then $A_i \in T_2$.

(2) If $\{A_{j_k}\}_{k=1}^{p} \in T_2$, where $j_1 < j_2 < \ldots < j_p$, then $A_\ell \in T_2$, if and only if,

(a) $\ell > j_p$,

and (b) $[A_\ell, A_{j_k}] \notin U$ where $k=1,\ldots,p$.

If $T_2=\emptyset$, we are finished. If not, we construct $T_3 \subset (X-\bigcup_1^2 T_i)$ in the same way as we constructed T_2.

Since X is finite, there will exist a least integer r such that $T_r=\emptyset$. We note also that $\bigcup_1^{r-1} T_i=X$ and that if $i \neq j$, then $T_i \cap T_j=\emptyset$. We can now assign a different color to each T_i. If $A_i, A_j \in T_k$, $i \neq j$, for some k, then $[A_i,A_j] \notin U$. Thus no adjacent nodes will have the same color. The algorithm furnishes an upper bound for the chromatic number k(G). Thus $k(G) \leq r-1$.

Welsh and Powell further state that

$$k(G) \leq r-1 \leq \max_i \min(i, d_i+1)$$

Proof of this upper bound comes from analysis of the algorithm. By construction, each node belonging to $(X-T_1)$ is adjacent to a node of T_1. By construction we see that each node of $(X-\bigcup_1^m T_j)$ is adjacent to at least one node of each of $\{T_j\}_1^m$. Hence, $A_k \in (X-\bigcup_1^m T_i)$ implies $d_k \geq m$.

Consider $(X-\bigcup_1^{d_i+1} T_j)$. A_i cannot belong to this set because $d_i \ngeq d_i+1$. Hence,

(1) $A_i \in \bigcup_1^{d_i+1} T_j$.

Since each T_j contains at least one element,

(2) $A_i \in \bigcup_1^{i} T_j$.

Together (1) and (2) above imply that

(3) $A_i \in \bigcup_{1}^{\varphi(i)} T_j$, where $\varphi(i)=\min(i,d_i+1)$.

Let $\delta(G)=\max_i \varphi(i)$. Then $A_i \in \bigcup_{1}^{\varphi(i)} T_j \subset \bigcup_{1}^{\delta(G)} T_j$ for all i. Setting $q=\delta(G)+1$ we see that $T_q=\emptyset$. Thus for the least integer r where $T_r=\emptyset$, this algorithm ensures $r \leq q$. Hence,

$k(G) \leq r-1 \leq q-1=\delta(G)=\max_i \min(i,d_i+1)$. The desired inequality is proved.

The above $\delta(G)$ provides us with an upper bound for the chromatic number of a graph. Berge's [2] statement of Brooks' [4] theorem on coloring the nodes of a network gives us insight about a lower bound. This statement, simplified to apply to our problem, is as follows:

Theorem (Brooks). A graph G of maximum degree q where $q>2$ can be colored with q colors except in the case where a component of G consists of q+1 vertices, each of which is joined to the other q vertices.

The above mentioned component is referred to as a complete subgraph of G. In other words, if G contains a complete subgraph on m vertices, then $k(G) \geq m$. Thus the largest complete subgraph of G sets a lower bound on k(G).

To illustrate the use of this algorithm in coloring, consider the given arbitrary graph in Figure 7 below.

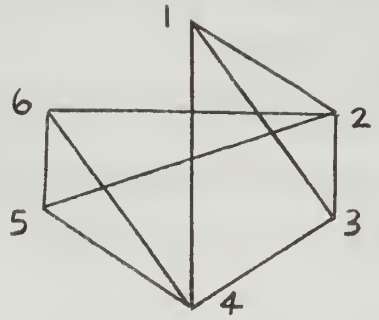


Fig. 7. An Arbitrary Graph

Numbering of the nodes must be done so that their degrees form a nonincreasing sequence. One such renumbering is listed below.

Node	Degree	Renumbered
1	3	A_3
2	4	A_1
3	3	A_4
4	4	A_2
5	3	A_5
6	3	A_6

Figure 8 below illustrates the arbitrary graph with nodes relabeled.

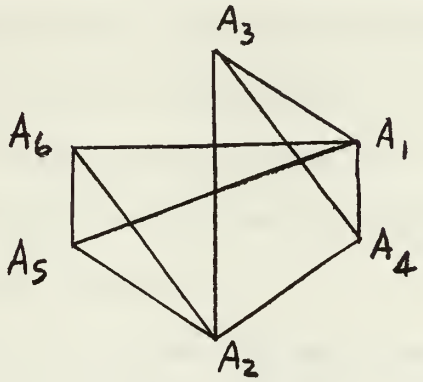


Fig. 8. Relabeled Arbitrary Graph

Using the algorithm to form the disjoint sets T_i, $i=1,.,4$, we see the following:

$T_1=\{A_1,A_2\}$, $T_2=\{A_3,A_5\}$, $T_3=\{A_4,A_6\}$, and $T_4=\emptyset$.

To obtain a satisfactory coloring, we assign the nodes of T_1, T_2, and T_3 colors 1, 2, and 3 respectively.

We note from the above that this arbitrary graph is 3 colorable by the algorithm. Of interest is a comparison of this number with the lower and upper bounds of the chromatic number for this graph. By observation, the

largest complete subgraph has 3 nodes; therefore, the lower bound is 3. Since the number of colors in our solution equals the lower bound, we deduce that the chromatic number is in fact 3. The upper bound of k(G), $\delta(G)=\max_i \min(i,d_i+1)=\max\{1,2,3,4,4,4\}$ is 4 for this example.

While in general we will not know what k(G) is for an arbitrary graph, this algorithm does ensure a p-coloring where $p \leq \delta(G)=\max_i \min(i,d_i+1)$.

B. Coloring the Graph by Color Vectors.

In coloring our conflict and sequence graphs we are restricted to five day colors and nine hour colors per day. Flowchart A in the Appendix illustrates the coloring procedure and application of the Welsh and Powell algorithm to this problem by means of a digital computer program using Fortran IV language. This procedure is described briefly in chronological order as follows:

(1) Section and instructor course assignments are read in.

(2) A course directory is established.

(3) An indexed list of conflicts is determined based on section and instructor conflicts.

(4) Course hour data is read in.

(5) Course hour nodes are given day colors using an arbitrary selection process.

(6) Course hour nodes for each day are given hour colors using the algorithm.

Data for the program has three forms. First are the sections cards containing the section, the number of students, the number of aviators for that section, and each course taken by that section. To differentiate between courses, segmentation and lab designations were used. To indicate instructor assignments, data cards were used with the same format as on section course assignment cards. This procedure simplified the computation of

conflicts between course hour nodes. Flowchart B in the Appendix illustrates the read-in and print-out of conflict data.

Flowchart C in the Appendix illustrates the computation of a course directory. Each course is associated with an integer, and the integer listing of these courses is used for all further computations. This course listing is printed out for reference. Included in this computation is a random number generator used for random reordering of this course directory. Flowchart I in the Appendix illustrates RAN the generator function used.

The list of conflicts is computed next. This is shown in Flowchart D in the Appendix. In the example described in Section 2, the directory assumes the following form.

CC(I)	I
COURS1	1
COURS2	2
COURS31	3
COURS32	4
COURS4	5
COURS4L	6

COURS31 and COURS32 are segments 1 and 2 respectively of COURS3 and COURS4L is the lab portion of COURS4. The conflict list for the above example is shown below.

J	1	2	3	4	5	6	7									
INDEX(J)	1	4	8	11	15	18	21									
I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
LIST(I)	2	3	4	1	3	5	6	1	2	4	1	3	5	6	2	...

The above information is interpreted as follows: COURS1 whose integer representation is 1 has conflicts with COURS2, COURS31, and COURS32, whose integer representations are 2, 3, and 4 respectively. The conflicts are found by entering the list at index(1) and reading all entries up to index(2), where the conflicting courses of COURS2 begin. This notation has the advantage of requiring less memory space and less print-out space than the comparable matrix representation discussed earlier. The nonzero elements of the I-th row or column of this matrix are listed starting at index(I).

The list which describes the conflicts between the nodes is ordered so that the nonzero elements corresponding to the I-th row or column of the matrix form an increasing sequence. Flowchart E in the Appendix describes this procedure and the print-out of its results. Flowchart J in the Appendix displays the subroutine RELIST used in this procedure.

Incidental to the main objective of the program, an upper bound for the chromatic number of the course graph is computed. During the development of the program it served to give an indication of the degree of difficulty

to be encountered later in the coloring phase. Flowchart F in the Appendix shows the computation of the upper bound for this graph using courses as nodes.

We next read course hour information into our program, in order that day colors may be assigned to the course hour nodes. Flowchart G in the Appendix illustrates this input and assignment of day colors. Flowchart K describes the subroutine DAYCOL called for assignment of day colors. This routine assigns recitation hours of the same course different day colors. Each lab course is recognized as such in order that its hours may be scheduled during the same day. A listing with index is compiled as shown, to remember these day color assignments. In our example the course hour information is as follows:

I	CC(I)	HOURS
1	COURS1	3 recitation
2	COURS2	3 recitation
3	COURS31	3 recitation
4	COURS32	3 recitation
5	COURS4	3 recitation
6	COURS4L	2 lab

Using the above course hour information DAYCOL assigned the day colors in our example as follows:

Day Color	1	2	3	4	5
Course	1	1	1	2	2
Hour	2	3	3	3	4
Nodes	4	4	5	5	5
	6				
	6				

To each day we now apply the coloring algorithm. Flowchart H in the Appendix gives the general procedure for the hour coloring. Flowchart L illustrates the HRCOLR subroutine used in that procedure. When labs having more than one period are scheduled on any given day, the integer representation for that course will be repeated for that day. To enable these nodes to maintain their separate identities, a renumbering of nodes is made. Relative to this new numbering, an indexed list of conflicts is computed from the master list for each particular day. Then the degree of each node is determined. Knowing this, the program assigns colors to each node according to the coloring procedure described earlier. The result is the second or hour component of a color vector. Having day colors already determined by DAYCOL, the process is complete.

Using the above program, data from the second quarter of the 1967-1968 academic year at the Naval Postgraduate School was processed. No special periods such as flight hours, instructor research time, or departmental

meetings were included in this data. Included were 424 courses, 403 sections, and 194 instructors. The number of conflicts for any course varied from 0 to 52; however, over half of the courses had conflicts ranging from 5 to 13.

The number of hour colors required for the five days were 9, 8, 10, 10, and 10. To investigate the effect of the DAYCOL assignments on the total number of hour colors, a random number generator was used to reorder the directory of courses prior to the DAYCOL routine in the program. Flowchart J in the Appendix illustrates this procedure for the given input data set. Table 1 displays the results of that investigation for 18 runs, including the first already mentioned. Each entry is the total number of hour colors for a particular run and day color.

None of the runs, which averaged about 7 minutes of IBM 360/67 time, gives the acceptable nine or less hour colors for all days. Column 6 of the table indicates the number of acceptable days for each run. Runs 3 and 18 were the best in this category having 3 each. Runs 1, 15, and 17 had 2 acceptable days each. All others had one or less. Column 7 of the table indicates the number of hour colors exceeding 9 each day, summed over the five days for each run.

4. Conclusions.

While no acceptable coloring solution has been found directly from the described program, several program runs are sufficiently close to indicate that modifications in the coloring technique may in fact give a satisfactory solution where an acceptable trial and error solution is known to exist. The known solution, hand generated from the same data with all constraints taken into account, used nine periods for each of the five days. Coloring runs 1 and 18, Table 1, both came within 3 periods or hour colors of equalling the number of periods required within one week. Runs 3, 12, and 17 were also noticeably close to a solution.

In this case where a solution is known to exist, the results obtained must be attributed to the coloring technique applied. With an acceptable solution in mind, we see that any coloring algorithm or technique which yields nine or less hour colors per day is a satisfactory procedure. Changes in the present program in DAYCOL or in the algorithm itself could improve results. One such change in the algorithm may be the following:

T_1 can be determined by the Welsh and Powell algorithm. The set of nodes in T_1 is then removed from the graph, together with the associated edges. The remaining subgraph is treated as a new graph, and the Welsh and Powell is reapplied to find the set T_2, etc.

Also a change, which should be considered, is a coloring program where the day and hour colors, dependent upon one another, are assigned together. This involves combining the functions of DAYCOL and HRCOLR, with whatever algorithm used, into a program operating on a total conflict graph rather than separate hourly and daily graphs.

Another observation concerning our schedule problem can be made. Given arbitrary data, we may question the existence of any schedule solution. A solution to the presently unsolved problem of determining the chromatic number for an arbitrary graph will indicate if a schedule solution is in fact possible.

BIBLIOGRAPHY

1. Appleby, J. S., D. V. Blake, and E. A. Newman. "Techniques for Producing School Timetables on a Computer and Their Application to other Scheduling Problems," Computer Journal, Vol. 3, 5 (Jan., 1961), 237-245.

2. Berge, Claude. "Graph Theory," American Mathematical Monthly, Vol. 71, 5 (May, 1964), 471-481.

3. Berge, Claude. The Theory of Graphs and its Applications. New York: John Wiley and Sons, Inc., 1964. 247 pp.

4. Brooks, R. L. "On Colouring the Nodes of a Network," Proceedings of the Cambridge Philosophical Society, Vol. 37, 1941, 194-197.

5. Cole, A. J. "The Preparation of Examination Timetables Using a Small-store Computer," Computer Journal, Vol. 7, 2 (July, 1964), 117-121.

6. Csima, J., and C. C. Gottlieb. "Tests on a Computer Method for Constructing School Timetables, Communications of the Association for Computing Materials, Vol 3 (March, 1964), 160-163.

7. Holz, Robert E. "State of the Art of Automatic Scheduling and Registration," Proceedings of the 9th Conference of College and University Records, 1964.

8. Ore, Oystein. "Theory of Graphs," American Mathematical Society Colloquium Publications, Vol. 38, 1962.

9. Sherman, Gordon R. "A Combinatorial Problem Arising from Scheduling University Classes," Journal of the Tennessee Academy of Science, Vol. 38, 3 (July, 1963), 115-117.

10. Welsh, D. J. A., and M. B. Powell. "An Upper Bound for the Chromatic Number of a Graph and its Applicability to Timetabling Problems," Computer Journal, Vol. 10, 1 (May, 1967), 85-86.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library Naval Postgraduate School Monterey, California 93940	2
3.	Office of Naval Research Attn: Dr. Adkins Department of the Navy Washington, D. C. 20350	1
4.	Associate Professor U. R. Kodres Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
5.	LT J. A. Mack, III, USN 4 New Town Lane, The Crescent Charleston, S. C. 29407	1

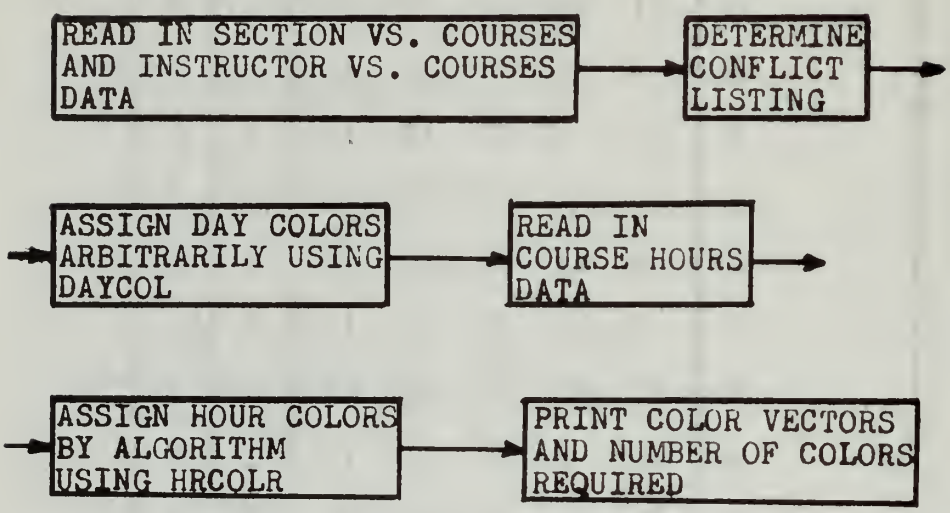
TABLE 1

Day Color		1	2	3	4	5	Col 6	Col 7
Runs	1	9	8	10	10	10	2	3
	2	11	10	10	11	10	0	7
	3	9	9	9	11	11	3	4
	4	11	9	11	10	10	1	6
	5	10	10	12	11	10	0	8
	6	9	12	10	12	10	1	8
	7	9	13	12	11	10	1	10
	8	12	10	10	10	11	0	8
	9	10	8	13	11	10	1	8
	10	10	10	10	12	10	0	7
	11	10	9	12	10	10	1	6
	12	10	9	10	11	10	1	5
	13	11	10	11	12	9	1	8
	14	11	12	11	9	11	1	9
	15	9	14	11	9	10	2	8
	16	12	11	12	9	11	1	10
	17	11	9	11	10	9	2	5
	18	10	9	9	11	9	3	3

APPENDIX

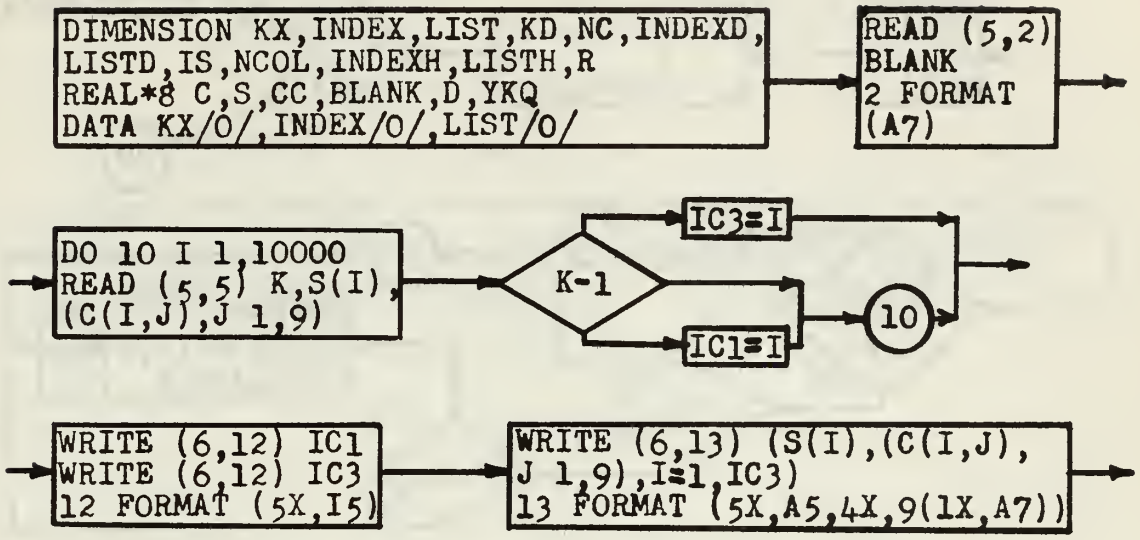
GENERAL COLORING PROCEDURE

FLOWCHART A



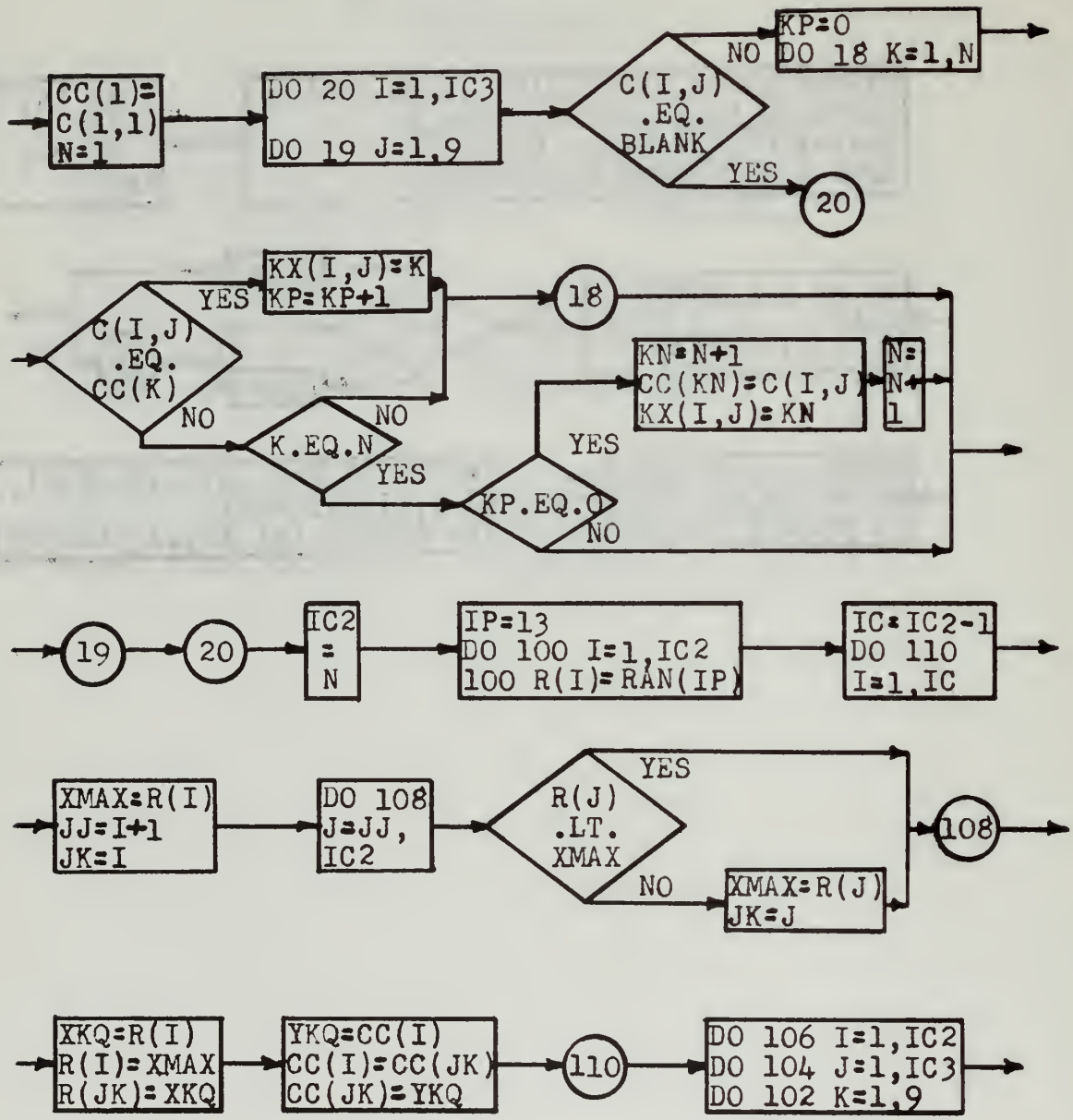
READING CONFLICT DATA

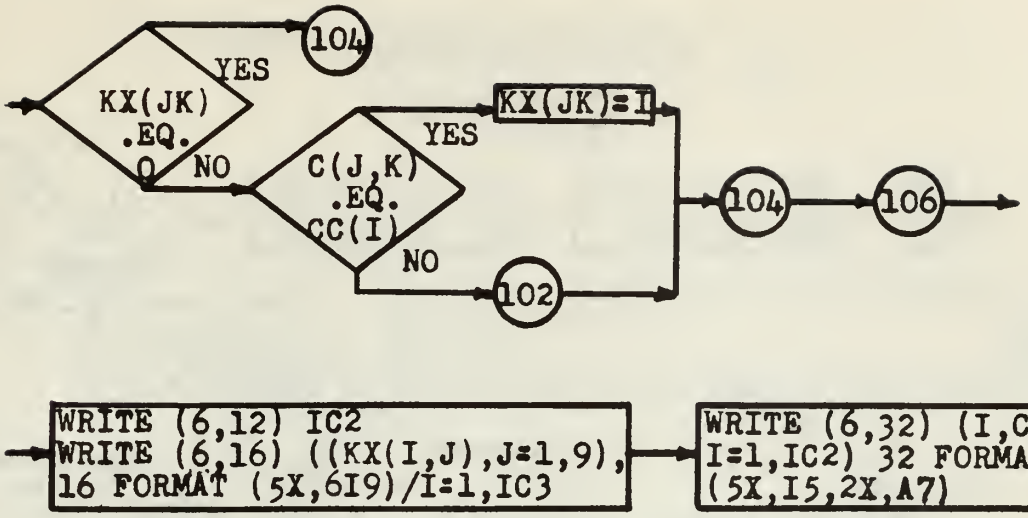
FLOWCHART B



COMPUTING COURSE LIST

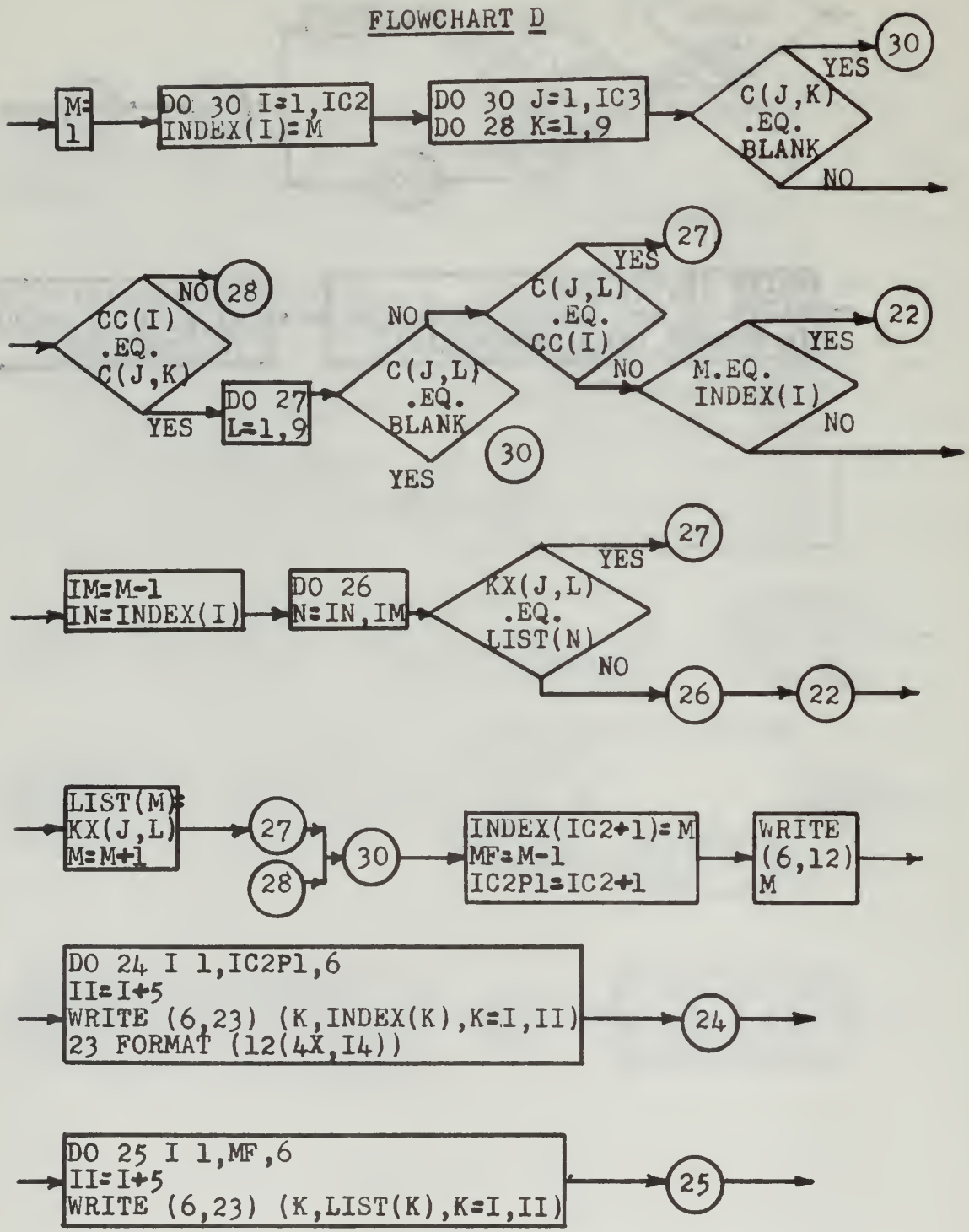
FLOWCHART C




104
YES
KX(JK)
.EQ.
0
NO
C(J,K)
.EQ.
CC(I)
YES
KX(JK)=I
NO
102
104
106
WRITE (6,12) IC2
WRITE (6,16) ((KX(I,J),J=1,9),
16 FORMAT (5X,6I9)/I=1,IC3
WRITE (6,32) (I,CC(I),
I=1,IC2) 32 FORMAT
(5X,I5,2X,A7)

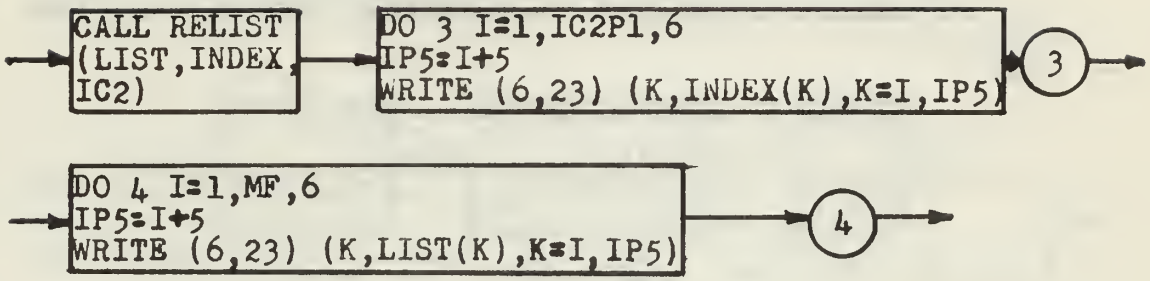
COMPUTING CONFLICT LISTING

FLOWCHART D



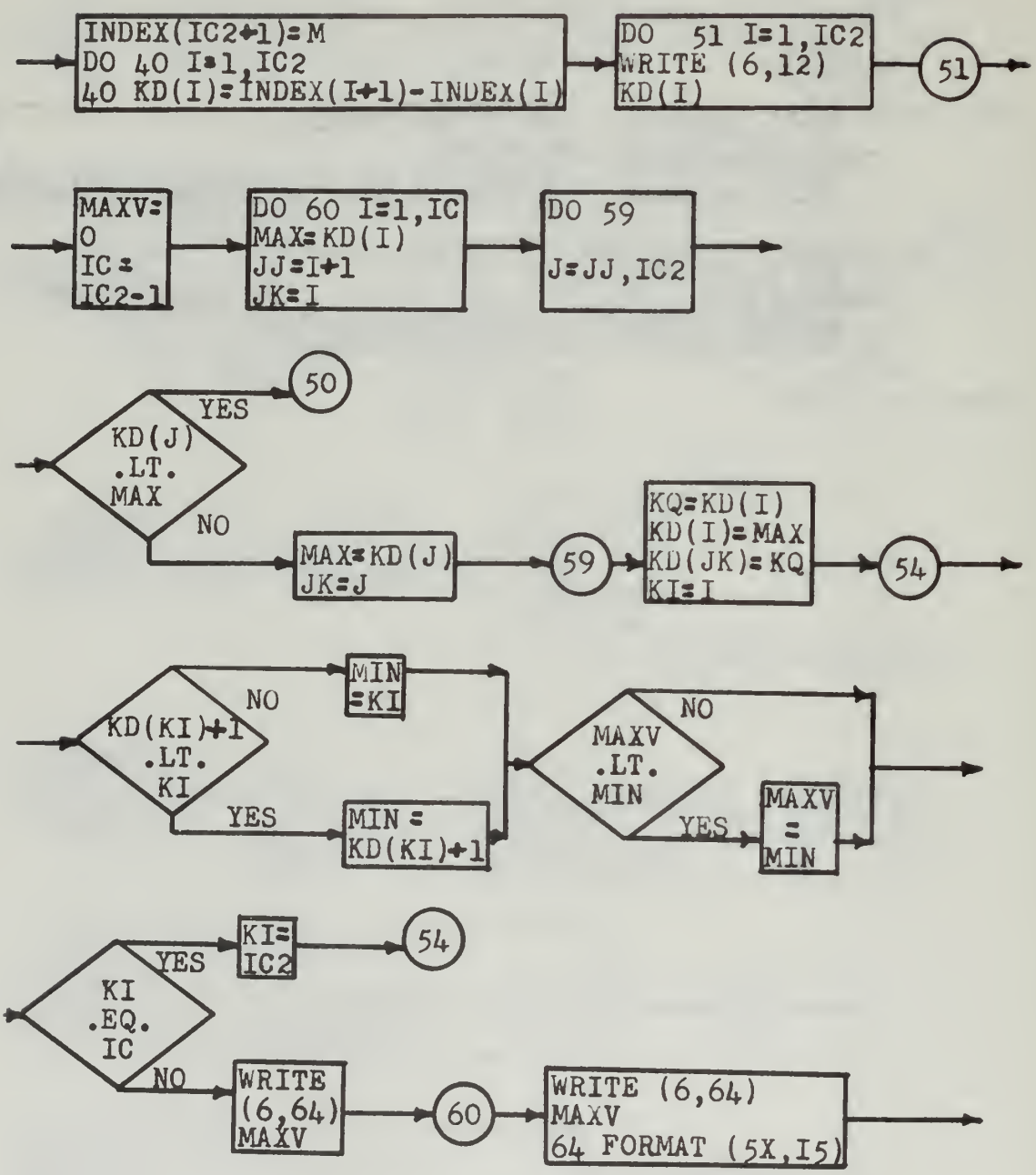
REORDERING CONFLICT LIST

FLOWCHART E



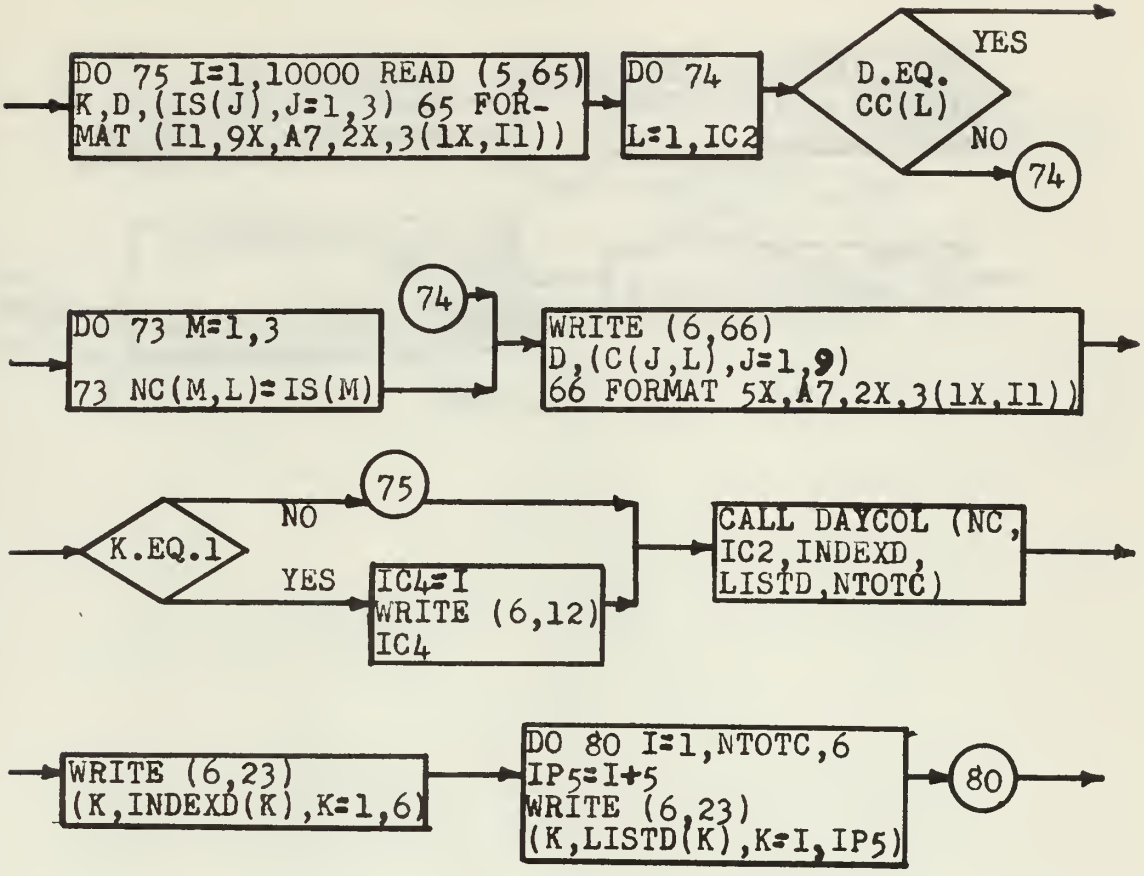
COMPUTING UPPER BOUND

FLOWCHART F



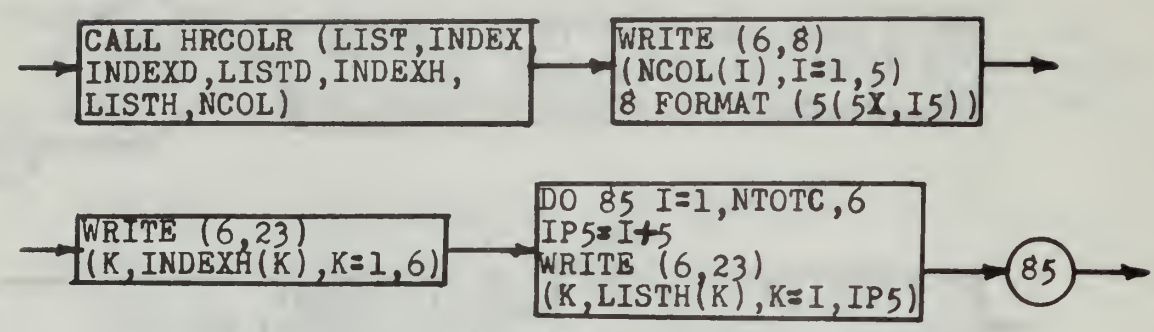
READING IN COURSE HOURS AND COMPUTING DAY COLORS

FLOWCHART G



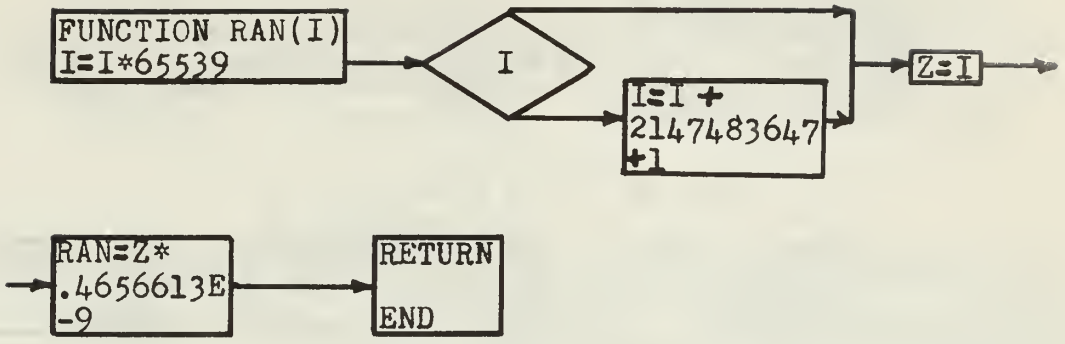
COMPUTING HOUR COLORS

FLOWGRAPH H



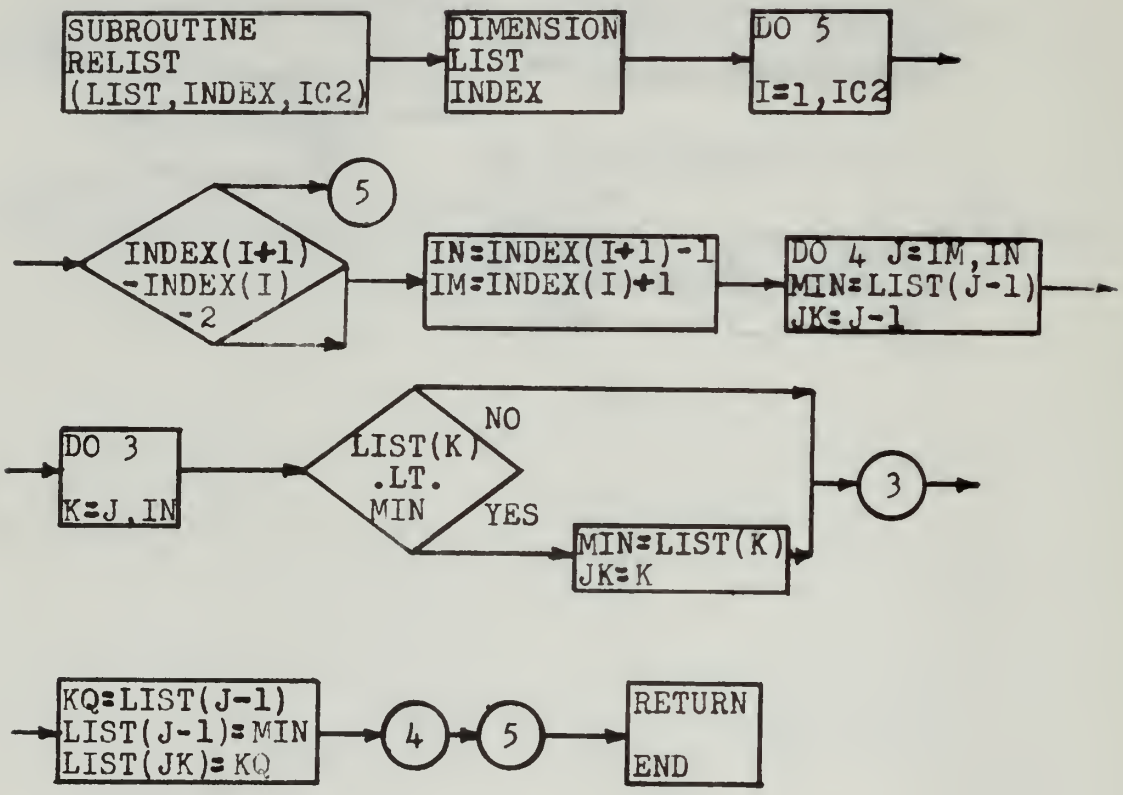
FUNCTION RAN(I)

FLOWCHART I



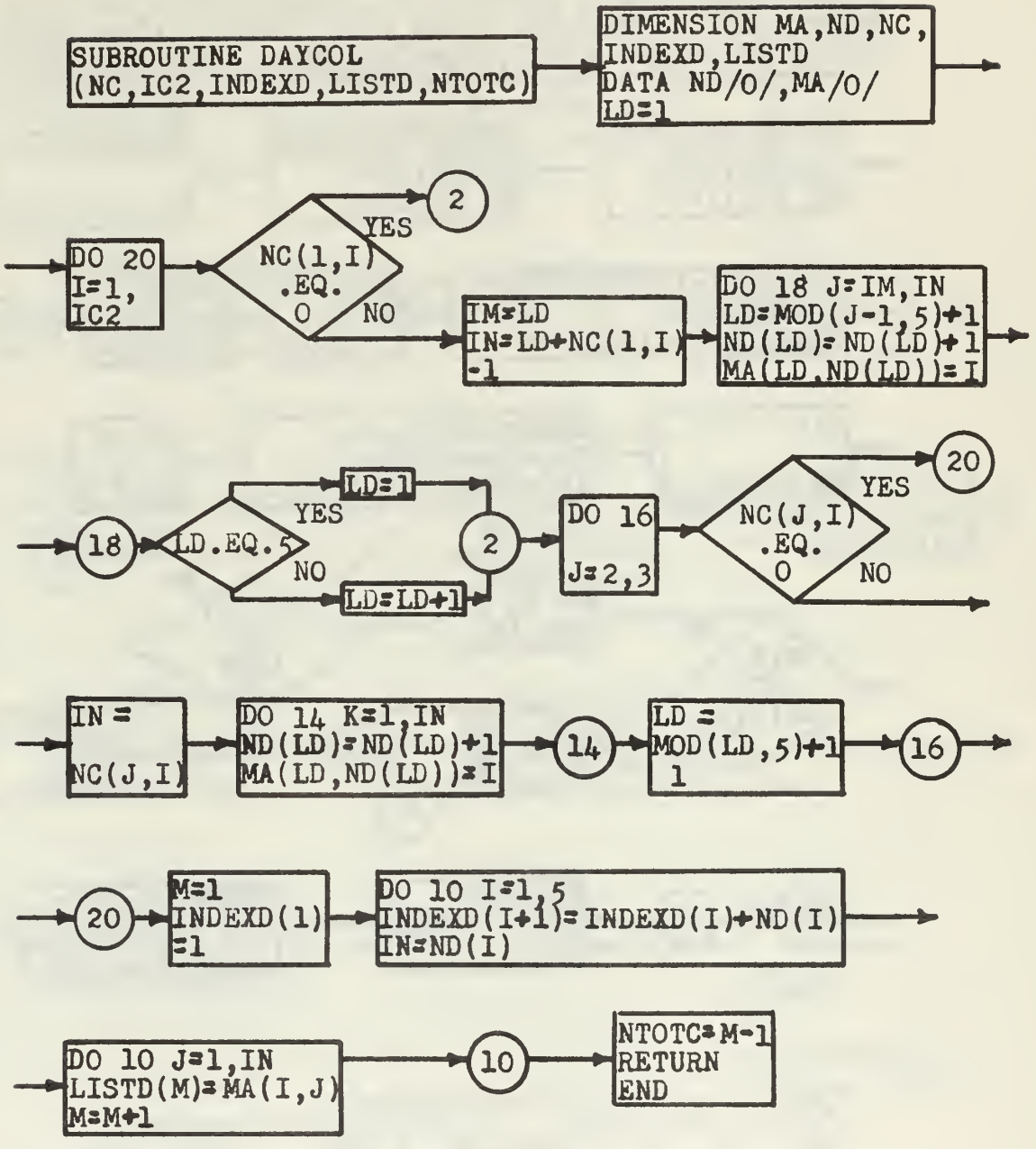
SUBROUTINE RELIST

FLOWCHART J



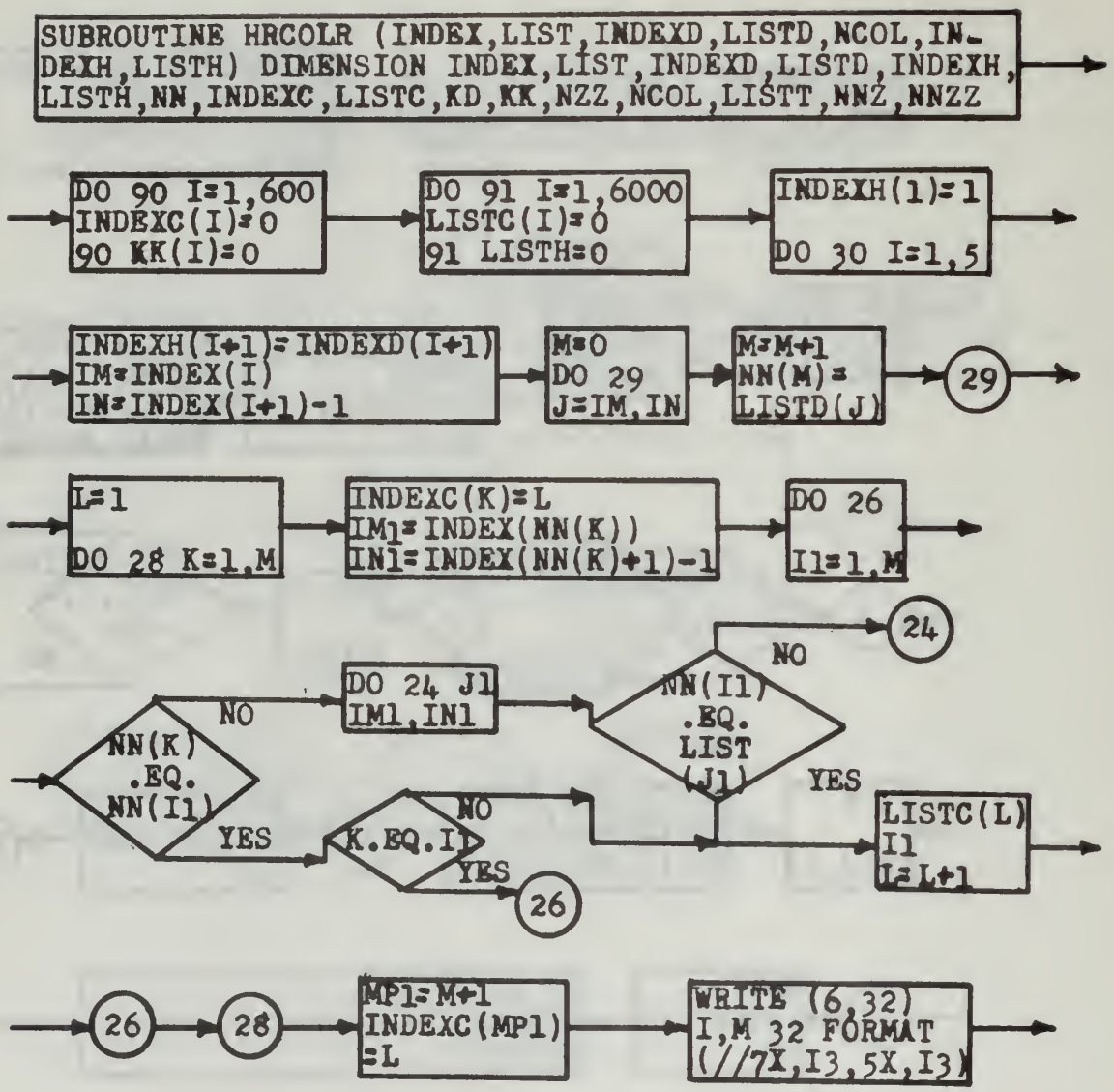
SUBROUTINE DAYCOL

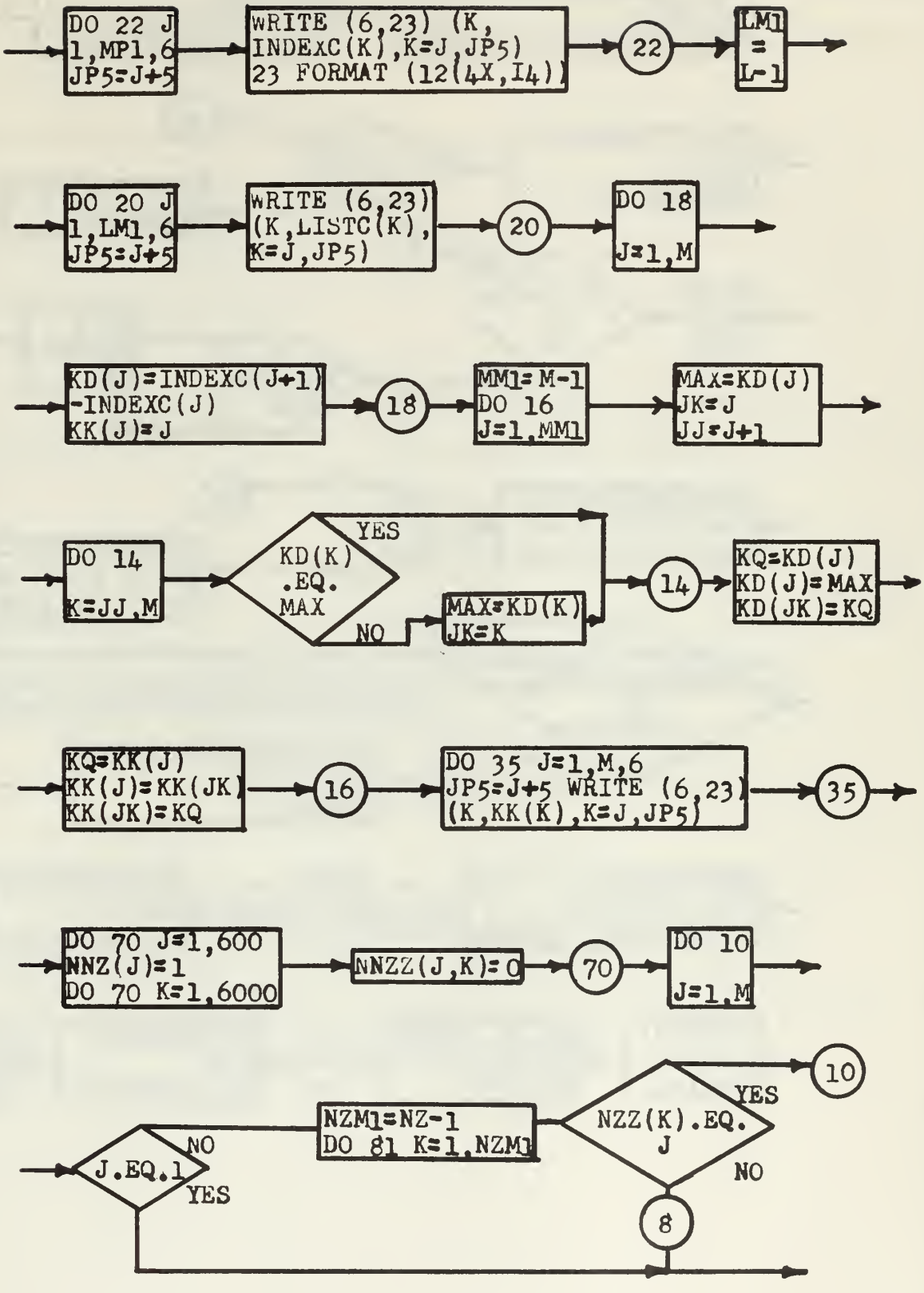
FLOWGRAPH K

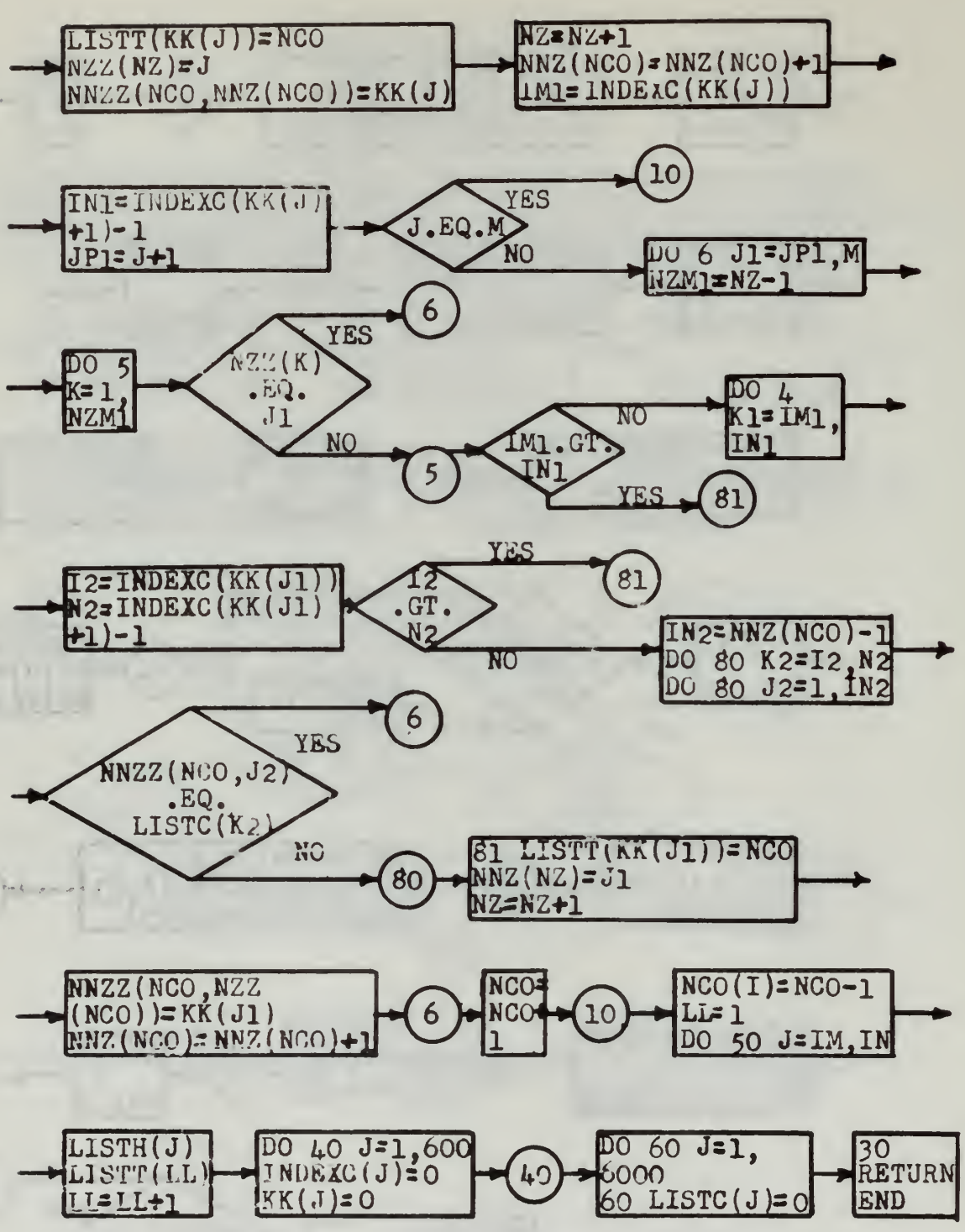


SUBROUTINE HRCOLR

FLOWCHART L




DO 22 J
1,MP1,6
JP5=J+5
WRITE (6,23) (K,
INDEXC(K),K=J,JP5)
23 FORMAT (12(4X,I4))
22
LM1
=
L-1
DO 20 J
1,LM1,6
JP5=J+5
WRITE (6,23)
(K,LISTC(K),
K=J,JP5)
20
DO 18
J=1,M
KD(J)=INDEXC(J+1)
-INDEXC(J)
KK(J)=J
18
MM1=M-1
DO 16
J=1,MM1
MAX=KD(J)
JK=J
JJ=J+1
DO 14
K=JJ,M
KD(K)
.EQ.
MAX
YES
NO
MAX=KD(K)
JK=K
14
KQ=KD(J)
KD(J)=MAX
KD(JK)=KQ
KQ=KK(J)
KK(J)=KK(JK)
KK(JK)=KQ
16
DO 35 J=1,M,6
JP5=J+5 WRITE (6,23)
(K,KK(K),K=J,JP5)
35
DO 70 J=1,600
NNZ(J)=1
DO 70 K=1,6000
NNZZ(J,K)=0
70
DO 10
J=1,M
10
NZM1=NZ-1
DO 81 K=1,NZM1
NZZ(K).EQ.
J
YES
NO
J.EQ.1
NO
YES
8


LISTT(KK(J))=NCO
NZZ(NZ)=J
NNZZ(NCO,NNZ(NCO))=KK(J)
NZ=NZ+1
NNZ(NCO)=NNZ(NCO)+1
IM1=INDEXC(KK(J))
IN1=INDEXC(KK(J)+1)-1
JP1=J+1
J.EQ.M
YES
NO
10
DO 6 J1=JP1,M
NZM1=NZ-1
DO 5 K=1,NZM1
NZZ(K).EQ.J1
YES
NO
6
5
IM1.GT.IN1
NO
YES
DO 4 K1=IM1,IN1
81
I2=INDEXC(KK(J1))
N2=INDEXC(KK(J1)+1)-1
I2.GT.N2
YES
NO
81
IN2=NNZ(NCO)-1
DO 80 K2=I2,N2
DO 80 J2=1,IN2
NNZZ(NCO,J2).EQ.LISTC(K2)
YES
NO
6
80
81 LISTT(KK(J1))=NCO
NNZ(NZ)=J1
NZ=NZ+1
NNZZ(NCO,NZZ(NCO))=KK(J1)
NNZ(NCO)=NNZ(NCO)+1
6
NCO=NCO+1
10
NCO(I)=NCO-1
LL=1
DO 50 J=IM,IN
LISTH(J)
LISTT(LL)
LL=LL+1
DO 40 J=1,6000
INDEXC(J)=0
KK(J)=0
40
DO 60 J=1,6000
60 LISTC(J)=0
30
RETURN
END

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

2b. GROUP

3 REPORT TITLE

An Application of Graph Coloring to a Scheduling Problem

4 DESCRIPTIVE NOTES *(Type of report and, inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

Jacob A. MACK, III

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
21 June 1968	56	10

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

[redacted]

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Naval Postgraduate School
Monterey, California 93940

13. ABSTRACT

A class scheduling problem is formulated as a graph coloring problem. A computer program based on an algorithm recently developed by Welsh and Powell, Computer Journal, Vol. 10, May 1967, pp. 85-86, is used to obtain a solution to the coloring problem. While the program fails to provide an acceptable schedule in this application, the results indicate that improvements in the coloring algorithm may yield acceptable schedules.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Unclassified

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Coloring						
Graph						
Scheduling						